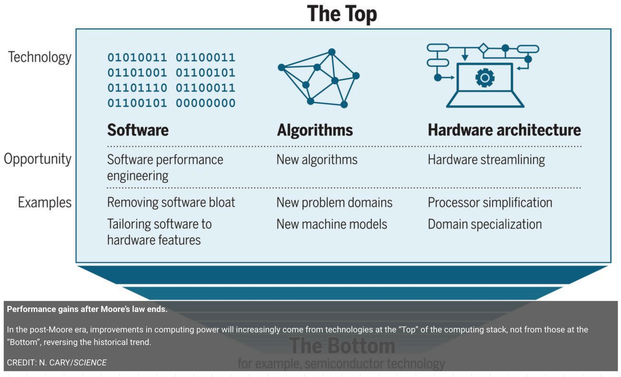
🚀 WebAssembly is the perfect runtime for server-side (or serverless) applications

• 3 minutes to read

In a recent paper on Science, MIT professor Leiserson et al. noted that, as Moore's Law stalled, *"practically all industries will face challenges to their productivity"*. But, while semiconductor miniaturization hits the quantum limit, the good news is that *"there's plenty of room at the top"*. The drivers for growth will be from more efficient software, new algorithms, and specialized hardware. The paper demonstrated that machine learning tasks can run 60,000x times faster when we replace Python with native code and specialized hardware. WebAssembly is a key technology that makes software more effcient, while preserving safety, portability, and software engineering best practices we built up in the past 25 years.



> Node.js and Rust developers get started here! Use high performance Rust functions in your Node.js apps. The Rust program is compiled into WebAssembly for safety, portability, and manageability in Node.js.

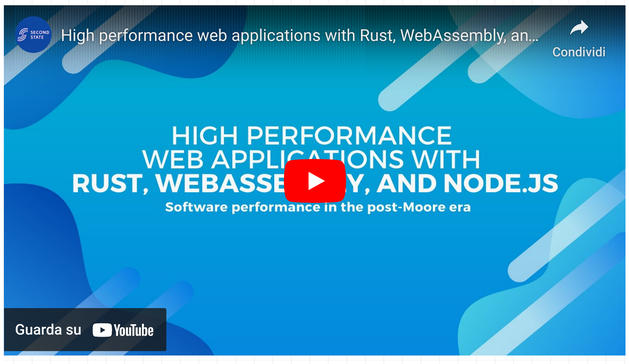
WebAssembly was invented as a client-side technology, but it is also proven very useful on the server-side. Server-side WebAssembly provides crucial benefits for modern web and service applications. It is

- Fast. WebAssembly achieves near native performance. Compared with the Java, Python or JavaScript runtimes, it can be 10x to 100x faster (how is this possible?). It is also much faster than Docker, especially in cold start and system access.
- Safe. WebAssembly is a sandbox with a capability-based security model. It is not only safer than native binaries, but also safer than OS-level containers like Docker. Yet it provides access to the underlying system, icnluding new hardware features.
- Portable. WebAssembly apps can be written in C, C++, Rust, Go, and run without change on different OS and hardware platforms.
- Manageable. WebAssembly programs can be provisioned, started, hot swapped, stopped, and moved around by other applications.

The major use cases for server-side WebAssembly are high-performance applications and untrusted code in the cloud.

Native code is often used where high performance and efficiency are required. The popular Node.js runtime is written in native C/C++. Native code runs AI inference, big data analytics, image and video processing, and scientific computing. WebAssembly is a light, fast, and cross-platform container. It is a safe and managed alternative to native code. Learn more here.

Untrusted code is the cornerstone of cloud native or serverless computing. For SaaS providers, users want to customize their experiences using code, or to create applications for their peers. Examples include Function as a Service, workflow apps, smart contracts, plugins, and extensions. WebAssembly supports multiple programming languages, and can provide a high performance sandbox for user submitted code with little resource consumption.



Second State provides an open-source WebAssembly implementation (Second State Virtual Machine, or SSVM) that is specifically optimized for server side applications. It is

- Best in-class in performance. It is 1000x faster than Docker for cold starts.
- Seamlessly supports server application frameworks, such as the Node.js. You can build high performance Node.js apps with SSVM.
- Supports safe access to external resources, such as databases, message queues, and even new AI hardware
- Allows precise metering of computational resources for serverless apps.

WebAssembly #question-answer #serverless #function-as-a-service #ai-as-a-service

SECOND STATE

A high-performance, extensible, and hardware optimized WebAssembly Virtual Machine for automotive, cloud, AI, and blockchain applications

second-state @secondstateinc